|  |
| --- |
| College LaSalle |
| Project - Oriented Object Programming User and Technical Manual |
|  |
| Presented to: Mihai Maftei. |

|  |
| --- |
| Bernadette Nicole Fernando  4/13/2023  Version: 3.0 |

1. **Start by adding a short description of your project, and the languages (technologies) used:**
2. Language: C#
3. Tools (IDE):

* Microsoft Visual Studio Community 2022 Version 17.5.2
* Microsoft .NET Framework Version 4.8.09032
* Installed Version: Community

1. **Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).**

|  |
| --- |
| **FORM:** DASHBOARD **NAME:** frmDashboard |
| 1. To navigate through the different forms available in the dashboard, click one of the tabs on top 2. To enter a form, simply click on one of the buttons under the tabs 3. To exit the application, click the exit button at the bottom right of the dashboard 4. A prompt will appear and will require the user to confirm before closing the application |

|  |
| --- |
| **FORM:** LOTTO 649 **NAME:** frm649 |
| 1. To generate winning numbers for Lotto 649, click the Generate button at the lower right part of the form 2. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created. 3. After thefile is created, another propmpt will appear to tell the user that the generated numbers are added successfully to the file 4. The results will be displayed on the form interface. The 6 winning numbers with 1 bonus are displayed on the read-only text box on the right and the random 7-digits are displayed on the bottom of the image 5. To view the Lotto file, click the Read File button in the middle below the form 6. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file 7. To exit form, click Exit button at the bottom right 8. A prompt will appear asking for confirmation from user before the application closes |

|  |
| --- |
| **FORM:** LOTTO MAX **NAME:** frmMax |
| 1. To generate winning numbers for Lotto Max, click the Generate button at the lower right part of the form 2. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created. 3. After thefile is created, another propmpt will appear to tell the user that the generated numbers are added successfully to the file 4. The results will be displayed on the form interface. The 7 winning numbers with 1 bonus are displayed on the read-only text box on the right and the random 7-digits are displayed on the bottom of the image 5. To view the Lotto file, click the Read File button in the middle below the form 6. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file 7. To exit form, click Exit button at the bottom right 8. A prompt will appear asking for confirmation from user before the application closes |

|  |
| --- |
| **FORM:** MONEY EXCHANGE **NAME:** frmMoneyEx |
| 1. To convert to and from any currency, the user should select the radio button beside the currency to convert FROM (by default, it is CAD) 2. Input the amount to convert in the textbox under the FROM group box 3. Next, the user should click the radio button beside the currency to convert TO (by default, it is CAD) 4. Click Convert button to convert the input FROM and TO the desired currencies 5. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created. 6. After the file is created, another propmpt will appear to tell the user that the output is added successfully to the file 7. The results will be displayed on the form interface. The converted currency will be displayed on the read-only textbox under the TO group box 8. To view the content of the text file, click the Read File button in the middle below the form 9. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file      1. To exit form, click Exit button at the bottom right 2. A prompt will appear asking for confirmation from user before the application closes. It will also display the total runtime of the form between the time it is opened and Exit button is clicked. |

|  |
| --- |
| **FORM:** TEMPERATURE CONVERT **NAME:** frmTempConv |
| 1. To convert temperature, the user must select between two conversion options: from C to F (converting from Celsius to Fahrenheit) or from F to C (converting from Fahrenheit to Celsius). The user should click the radio button beside the desired conversion. By default, from C to F is selected.      1. The user must input the value to be converted 2. To convert the temperature on the left, click the Convert button on the lower left bottom pf the form 3. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created. 4. After the file is created, another propmpt will appear to tell the user that the output is added successfully to the file 5. The results will be displayed on the form interface. The converted temperature will be displayed on the read-only textbox on the right.The temperature description is also displayed on the read-only textbox with appropriate colors. 6. To view the content of the text file, click the Read File button in the middle below the form 7. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file      1. To exit form, click Exit button at the bottom right 2. A prompt will appear asking for confirmation from user before the application closes. |

|  |
| --- |
| **FORM:** CALCULATOR **NAME:** frmCalculator |
| 1. To use the calculator, click the number buttons to build the first number 2. Then ,click the desired operation 3. Then, click the number buttons to build the second number 4. To view the result, there can be two ways: Clicking the equal button or clicking the next operation. Screenshot below shows the result when Equal button is clicked. 5. When the user clicks the next operation, the result of the previous two numbers will be the first number and the user can once again click the number buttons for the second number. Screenshot below shows the result when Operator button is clicked. 6. After the second number is entered, clicking any of the buttons (+,-,\*,/,=) on the right of the number buttons will display the result of the operations. 7. To clear or reset the calculator, click the Clear button on the lower left of the form 8. To exit form, click Exit button at the bottom right 9. A prompt will appear asking for confirmation from user before the application closes. |

|  |
| --- |
| **FORM:** IPV4 VALIDATOR **NAME:** frmIPv4Validator |
| 1. To validate an IP address, type a valid IPv4 address on the textbox then click the Validate button on the lower left side of the form 2. If it’s a valid IPv4 address, a prompt confirming the entry will appear. 3. If the binary file does not exist, a prompt will appear to tell the user that the binary file is not found and will be created. 4. After the file is created, another propmpt will appear to tell the user that the output is added successfully to the file 5. If the user inputs an invalid IP, a prompt will appear with a list of criteria for a valid IPv4 address 6. To reset the form, click the Reset button in the middle 7. To exit form, click Exit button at the bottom right 8. A prompt will appear asking for confirmation from user before the application closes. It will also display the total runtime of the form between the time it is opened and Exit button is clicked. |

|  |
| --- |
| **FORM:** DISPLAY READ FILE **NAME:** frmReadFile |
| 1. The read form serves as a display for the outputs of the other forms. It displays the output on a read-only textbox 2. To close the form, simply click the Exit button on the lower right |

1. **Present the code of your application (forms).**

|  |
| --- |
| **FORM:** DASHBOARD **NAME:** frmDashboard |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  // Bernadette Fernando  // April 04 2023  // This windows form will provide access to different forms that are able to do the following:  // Generate Numbers, Convert Currency, Calculator, Validate IPv4  namespace WinFormProject  {  public partial class frmDashboard : Form  {  public frmDashboard()  {  InitializeComponent();  }  private void btnLottoMax\_Click(object sender, EventArgs e)  {  frmMax app = new frmMax();  app.Show();  }  private void btnLotto649\_Click(object sender, EventArgs e)  {  frm649 app = new frm649();  app.Show();  }  private void btnMoneyEx\_Click(object sender, EventArgs e)  {  frmMoneyEx app = new frmMoneyEx();  app.Show();  }  private void btnTempConv\_Click(object sender, EventArgs e)  {  frmTempConv app = new frmTempConv();  app.Show();  }  private void btnCalc\_Click(object sender, EventArgs e)  {  frmCalculator app = new frmCalculator();  app.Show();  }  private void btnIPVal\_Click(object sender, EventArgs e)  {  frmIPv4Validator app = new frmIPv4Validator();  app.Show();  }  private void btnMainExit\_Click(object sender, EventArgs e)  {  if (MessageBox.Show("Do you want to quit this application? ", "Exit ?", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  Application.Exit();  }  }  }  } |

|  |
| --- |
| **FORM:** LOTTO 649 **NAME:** frm649 |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace WinFormProject  {  public partial class frm649 : Form  {  public frm649()  {  InitializeComponent();  }  private void btnL649Gen\_Click(object sender, EventArgs e)  {  // BASIC FUNCTION  // clear the textbox if button is clicked  txtL649.Clear();  var generatedNum = new List<int>();  Random random = new Random();  // generate random numbers from 0 to 9  string randomNumber = random.Next(0, 9).ToString();  for (int i = 0; i < 6; i++)  {  randomNumber += random.Next(0, 9).ToString();  }  lbl64909.Text = randomNumber;  // generate first number and add to list  int firstRNum = random.Next(1, 49);  generatedNum.Add(firstRNum);  // generate a new number and add to list  int newRNum;  for (int i = 0; i < 6; i++)  {    // generate a new number while it exists in the list  // validation to ensure that every number is unique  do  {  newRNum = random.Next(1, 49);  } while (generatedNum.Contains(newRNum));    generatedNum.Add(newRNum);  }  // store the last generated number as bonus  var lastItem = generatedNum.Last();  // display list  foreach (int num in generatedNum)  {  txtL649.Text += num.ToString() + "\r\n";  }  // WRITE TO TEXT FILE  string outputDesc = $"{",",-5}";  for (int i = 0; i < 6; i++)  {  // position the commas  outputDesc += generatedNum[i];  if (i < 5)  {  outputDesc += ",";  }  }  outputDesc += $"{"Bonus",8} {lastItem}";  DataStream toWrite = new DataStream();  toWrite.FileName = "LottoNbrs";  toWrite.MsgBoxTitle = "Lotto";  toWrite.Output = "649";  toWrite.Description = outputDesc;  toWrite.WriteFile();  }  private void btn649Exit\_Click(object sender, EventArgs e)  {  if (MessageBox.Show("Do you want to close this window? ", "Close Lotto 649", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  private void btnL649Read\_Click(object sender, EventArgs e)  {  // READ TEXT FILE  DataStream toRead = new DataStream();  toRead.FileName = "LottoNbrs";  toRead.MsgBoxTitle = "Lotto649";  frmReadFile readDisplay = new frmReadFile();  readDisplay.fileOutput = toRead.ReadFile();  readDisplay.frmTitle = toRead.MsgBoxTitle;  readDisplay.Show();  }  private void frm649\_Load(object sender, EventArgs e)  {  }  }  } |

|  |
| --- |
| **FORM:** LOTTO MAX **NAME:** frmMax |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Reflection;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  // Bernadette Fernando  // Windows Form Project  // April 18, 2023  namespace WinFormProject  {  public partial class frmMax : Form  {  public frmMax()  {  InitializeComponent();  }  private void btnLMaxGen\_Click(object sender, EventArgs e)  {  // BASIC FUNCTION  // clear the textbox if button is clicked  txtLMax.Clear();  var generatedNum = new List<int>();  Random random = new Random();  // generate random numbers from 0 to 9  string randomNumber = random.Next(0, 9).ToString();  for (int i = 0; i < 6; i++)  {  randomNumber += random.Next(0, 9).ToString();  }  lblLMax09.Text = randomNumber;  // generate first number and add to list  int firstRNum = random.Next(1, 50);  generatedNum.Add(firstRNum);  // generate a new number and add to list  int newRNum;  for (int i = 0; i < 7; i++)  {  // generate a new number while it exists in the list  // validation to ensure that every number is unique  do  {  newRNum = random.Next(1, 50);  } while (generatedNum.Contains(newRNum));  generatedNum.Add(newRNum);  }  // store the last generated number as bonus  var lastItem = generatedNum.Last();  foreach (int num in generatedNum)  {  txtLMax.Text += num.ToString() + "\r\n";  }    // WRITE TO TEXT FILE  string outputDesc = $"{",",-5}";  for (int i = 0; i < 7; i++)  {  // position the commas  outputDesc += generatedNum[i];  if (i < 6)  {  outputDesc += ",";  }  }  outputDesc += $"{"Bonus",8} {lastItem}";  DataStream toWrite = new DataStream();  toWrite.FileName = "LottoNbrs";  toWrite.MsgBoxTitle = "Lotto";  toWrite.Output = "Max";  toWrite.Description = outputDesc;  toWrite.WriteFile();  }  private void btnLMaxExit\_Click(object sender, EventArgs e)  {  if (MessageBox.Show("Do you want to close this window? ", "Close LottoMax", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  private void btnLMaxRead\_Click(object sender, EventArgs e)  {  // READ TEXT FILE  DataStream toRead = new DataStream();  toRead.FileName = "LottoNbrs";  toRead.MsgBoxTitle = "LottoMax";  frmReadFile readDisplay = new frmReadFile();  readDisplay.fileOutput = toRead.ReadFile();  readDisplay.frmTitle = toRead.MsgBoxTitle;  readDisplay.Show();  }  }  } |

|  |
| --- |
| **FORM:** MONEY EXCHANGE **NAME:** frmMoneyEx |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows.Forms;  // March 16, 2023  // Money Conversion Factors (based on 1CAD):  // Source: Royal Bank of Canada  // 0.71230 USD  // 0.66609 EUR  // 0.58333 GBP  // 38.22630 PHP  namespace WinFormProject  {  public partial class frmMoneyEx : Form  {  public frmMoneyEx()  {  InitializeComponent();  }  DateTime openFormTime, closeFormTime;  string currencyFrom, currencyTo;  // accept only amount from 0 to 99999.99  Regex validCurrency = new Regex(@"^([0-9]{1,5})(\.[0-9]{1,2})?$");  private void btnMEx\_Click(object sender, EventArgs e)  {  // BASIC FUNCTION  CurrencyEx moneyConvert = new CurrencyEx();  // FROM Group  if (optUSD.Checked)  {  currencyFrom = "USD";  }  else if (optEUR.Checked)  {  currencyFrom = "EUR";  }  else if (optGBP.Checked)  {  currencyFrom = "GBP";  }  else if (optPHP.Checked)  {  currencyFrom = "PHP";  }  else if (optCAD.Checked)  {  currencyFrom = "CAD";  }  // TO Group  if (optToUSD.Checked)  {  currencyTo = "USD";  }  else if (optToEUR.Checked)  {  currencyTo = "EUR";  }  else if (optToGBP.Checked)  {  currencyTo = "GBP";  }  else if (optToPHP.Checked)  {  currencyTo = "PHP";  }  else if (optToCAD.Checked)  {  currencyTo = "CAD";  }  try  {  if (validCurrency.IsMatch(txtConvFrom.Text.Trim()))  {  moneyConvert.Amt = Convert.ToDecimal(txtConvFrom.Text.Trim());  // convert amount to CAD  decimal convCAD = moneyConvert.ConvertToCAD(currencyFrom);  // convert CAD to any amount  decimal convCurrency = moneyConvert.ConvertToCurrency(convCAD, currencyTo);  txtConvTo.Text = convCurrency.ToString();  // WRITE TO TEXT FILE  DataStream toWrite = new DataStream();  toWrite.FileName = "MoneyConv";  toWrite.MsgBoxTitle = "Money Conversion";  toWrite.Output = $"{moneyConvert.Amt} {currencyFrom} = {convCurrency} {currencyTo}";  toWrite.Description = "";  toWrite.WriteFile();  }  else  {  MessageBox.Show($"Error! Please input valid amount format", "Invalid Input");  txtConvFrom.Text = "";  txtConvTo.Text = "";  txtConvFrom.Focus();  }  }  catch (Exception err)  {  MessageBox.Show($"Error! {err.Message}", "Exception Error");  txtConvFrom.Text = "0";  txtConvFrom.Focus();  }  }  private void btnMExRead\_Click(object sender, EventArgs e)  {  // READ TEXT FILE  DataStream toRead = new DataStream();  toRead.FileName = "MoneyConv";  toRead.MsgBoxTitle = "Money Conversion";  frmReadFile readDisplay = new frmReadFile();  readDisplay.fileOutput = toRead.ReadFile();  readDisplay.frmTitle = toRead.MsgBoxTitle;  readDisplay.Show();  }  private void frmMoneyEx\_Load(object sender, EventArgs e)  {  optCAD.Checked = true;  optToCAD.Checked = true;  txtConvFrom.Text = "0";  txtConvTo.Text = "0";  openFormTime = DateTime.Now;  }  private void btnMExExit\_Click(object sender, EventArgs e)  {  closeFormTime = DateTime.Now;  TimeSpan totRunTime = closeFormTime.Subtract(openFormTime);  if (MessageBox.Show("Do you want to close this window?\n" +  $"Total runtime: {totRunTime.Minutes} mins and {totRunTime.Seconds} seconds", "Close Money Exchange", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  }  } |

|  |
| --- |
| **FORM:** TEMPERATURE CONVERT **NAME:** frmTempConv |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace WinFormProject  {  public partial class frmTempConv : Form  {  public frmTempConv()  {  InitializeComponent();  }  private void btnTConv\_Click(object sender, EventArgs e)  {  ConvertTemp tempConvert = new ConvertTemp();  decimal convTemp;  // accept only temperaturatures between -999.99 to 999.99  Regex validTemp = new Regex(@"^(\-?\d{1,3})(\.\d{1,2})?$");  try  {  string inputTemp = txtTempFrom.Text.Trim();  if (validTemp.IsMatch(inputTemp))  {  tempConvert.Temp = Convert.ToDecimal(inputTemp);  char unitFrom, unitTo;  if (optCtoF.Checked)  {  // convert temp from C to F  convTemp = tempConvert.ConvertToFarenheit();  txtTempTo.Text = convTemp.ToString();  unitFrom = 'C';  unitTo = 'F';  // change text color  if (tempConvert.Temp >= 40)  {  rtfTempDesc.ForeColor = Color.Red;  }  else if (tempConvert.Temp >= 21 && tempConvert.Temp < 40)  {  rtfTempDesc.ForeColor = Color.Green;  }  else if (tempConvert.Temp < 21)  {  rtfTempDesc.ForeColor = Color.Blue;  }  }  else  {  // convert temp from F to C  convTemp = tempConvert.ConvertToCelcius();  txtTempTo.Text = convTemp.ToString();  unitFrom = 'F';  unitTo = 'C';  // change text color  if (tempConvert.Temp >= 104)  {  rtfTempDesc.ForeColor = Color.Red;  }  else if (tempConvert.Temp >= 70 && tempConvert.Temp < 104)  {  rtfTempDesc.ForeColor = Color.Green;  }  else if (tempConvert.Temp < 70)  {  rtfTempDesc.ForeColor = Color.Blue;  }  }  // display temp description  string tempDescription = tempConvert.DescribeTemperature(unitFrom);  rtfTempDesc.Text = tempDescription;  // WRITE TO TEXT FILE  DataStream toWrite = new DataStream();  toWrite.FileName = "TempConv";  toWrite.MsgBoxTitle = "Temperature Conversion";  toWrite.Output = $"{inputTemp} {unitFrom} = {convTemp} {unitTo}";  // remove line break before saving to text file  tempDescription = tempDescription.Replace("\n", "").Replace("\r", " ");  toWrite.Description = " " + tempDescription;  toWrite.WriteFile();  }  else  {  MessageBox.Show($"Error! Please input valid temperature format", "Invalid Input");  rtfTempDesc.Text = "";  txtTempFrom.Text = "";  txtTempTo.Text = "";  txtTempFrom.Focus();  }  }  catch (Exception err)  {  MessageBox.Show($"Error! {err.Message}", "Exception Error");  txtTempFrom.Text = "0";  rtfTempDesc.Text = "";  txtTempFrom.Focus();  }  }  private void optFtoC\_CheckedChanged(object sender, EventArgs e)  {  lblFrom.Text = "F";  lblTo.Text = "C";  }  private void optCtoF\_CheckedChanged(object sender, EventArgs e)  {  lblFrom.Text = "C";  lblTo.Text = "F";  }  private void frmTempConv\_Load(object sender, EventArgs e)  {  optCtoF.Checked = true;  optFtoC.Checked = false;  txtTempFrom.Text = "0";  txtTempTo.Text = "0";  rtfTempDesc.Text = "";  }  private void btnTConvRead\_Click(object sender, EventArgs e)  {  // READ TEXT FILE  DataStream toRead = new DataStream();  toRead.FileName = "TempConv";  toRead.MsgBoxTitle = "Temperature Conversion";  frmReadFile readDisplay = new frmReadFile();  readDisplay.fileOutput = toRead.ReadFile();  readDisplay.frmTitle = toRead.MsgBoxTitle;  readDisplay.Show();  }  private void btnTConvExit\_Click(object sender, EventArgs e)  {  if (MessageBox.Show("Do you want to close this window? ", "Close Temp Converter", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  }  } |

|  |
| --- |
| **FORM:** CALCULATOR **NAME:** frmCalculator |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Diagnostics;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using static System.Windows.Forms.VisualStyles.VisualStyleElement.ProgressBar;  namespace WinFormProject  {  public partial class frmCalculator : Form  {  public frmCalculator()  {  InitializeComponent();  }  private void frmCalculator\_Load(object sender, EventArgs e)  {  toWrite.MsgBoxTitle = "Calculator";  }  // initiate constructors  // Calculator - handle calculations  Calculator calculate = new Calculator();  // Datastream - handle read and write to text file  DataStream toWrite = new DataStream();  // variable to build the number  private string numBuilder = "";  // list that will hold the number inputs and results  public List<double> listNumbers = new List<double>();  // set the state of the buttons  private bool btnAddClicked = false;  private bool btnSubClicked = false;  private bool btnMulClicked = false;  private bool btnDivClicked = false;  private bool btnEqualClicked = false;  private bool btnDecClicked = false;  double res;  // number buttons  private void btn1\_Click(object sender, EventArgs e)  {  // if equal btn is already clicked, reset the string before input of new number  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "1";  txtCalc.Text = numBuilder;  }  private void btn2\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "2";  txtCalc.Text = numBuilder;  }  private void btn3\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "3";  txtCalc.Text = numBuilder;  }  private void btn4\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "4";  txtCalc.Text = numBuilder;  }  private void btn5\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "5";  txtCalc.Text = numBuilder;  }  private void btn6\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "6";  txtCalc.Text = numBuilder;  }  private void btn7\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "7";  txtCalc.Text = numBuilder;  }  private void btn8\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "8";  txtCalc.Text = numBuilder;  }  private void btn9\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "9";  txtCalc.Text = numBuilder;  }  private void btn0\_Click(object sender, EventArgs e)  {  if (btnEqualClicked)  {  numBuilder = "";  btnEqualClicked = false;  }  numBuilder += "0";  txtCalc.Text = numBuilder;  }  private void btnDecimal\_Click(object sender, EventArgs e)  {  // make sure that decimal is not clicked more than once  if (!btnDecClicked)  {  if (btnEqualClicked)  {  numBuilder = "0";  btnEqualClicked = false;  }  numBuilder += ".";  txtCalc.Text = numBuilder;  btnDecClicked = true;  }  }  // operation buttons  private void btnAdd\_Click(object sender, EventArgs e)  {  try  {  // store the value in the number builder  double num = Convert.ToDouble(numBuilder);  // add the number to list  listNumbers.Add(num);  // reset the number builder  numBuilder = "";  // if equal sign is not clicked, perform operation on first and second item in number list  // then reset the state of operator buttons  if (!btnEqualClicked)  {  if (btnSubClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnSubClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnSubClicked = false;  }  else if (btnMulClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnMulClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} \* {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnMulClicked = false;  }  else if (btnDivClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnDivClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnDivClicked = false;  }  else if (btnAddClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnAddClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnAddClicked = false;  }  }  // if equal sign is clicked  // from index 1, remove 2 items (values added by equal btn event)  // display the last value of result  else if (btnEqualClicked)  {  listNumbers.RemoveRange(1, 2);  txtCalc.Text = res.ToString();  // reset the state of all btns  btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;  }  // change state when btn is clicked  btnAddClicked = true;  }  catch (Exception)  {  MessageBox.Show("Error! Please input a number.", "Invalid Input");  }  }  private void btnSub\_Click(object sender, EventArgs e)  {  try  {  double num = Convert.ToDouble(numBuilder);  listNumbers.Add(num);  numBuilder = "";  if (!btnEqualClicked)  {  if (btnSubClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnSubClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnSubClicked = false;  }  else if (btnMulClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnMulClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} \* {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnMulClicked = false;  }  else if (btnDivClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnDivClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnDivClicked = false;  }  else if (btnAddClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnAddClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnAddClicked = false;  }  }  else if (btnEqualClicked)  {  listNumbers.RemoveRange(1, 2);  txtCalc.Text = res.ToString();  btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;  }  btnSubClicked = true;  }  catch (Exception)  {  MessageBox.Show("Error! Please input a number.", "Invalid Input");  }  }  private void btnMul\_Click(object sender, EventArgs e)  {  try  {  double num = Convert.ToDouble(numBuilder);  listNumbers.Add(num);  numBuilder = "";  if (!btnEqualClicked)  {  if (btnSubClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnSubClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnSubClicked = false;  }  else if (btnMulClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnMulClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} \* {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnMulClicked = false;  }  else if (btnDivClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnDivClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnDivClicked = false;  }  else if (btnAddClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnAddClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnAddClicked = false;  }  }  else if (btnEqualClicked)  {  listNumbers.RemoveRange(1, 2);  txtCalc.Text = res.ToString();  btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;  }  btnMulClicked = true;  }  catch (Exception)  {  MessageBox.Show("Error! Please input a number.", "Invalid Input");  }  }  private void btnDiv\_Click(object sender, EventArgs e)  {  try  {  double num = Convert.ToDouble(numBuilder);  listNumbers.Add(num);  numBuilder = "";  if (!btnEqualClicked)  {  if (btnSubClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnSubClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnSubClicked = false;  }  else if (btnMulClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnMulClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} \* {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnMulClicked = false;  }  else if (btnDivClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnDivClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnDivClicked = false;  }  else if (btnAddClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Solve(listNumbers, "btnAddClicked");  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  btnAddClicked = false;  }  }  else if (btnEqualClicked)  {  listNumbers.RemoveRange(1, 2);  txtCalc.Text = res.ToString();  btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;  }  btnDivClicked = true;  }  catch (Exception)  {  MessageBox.Show("Error! Please input a number.", "Invalid Input");  }  }  private void btnEqual\_Click(object sender, EventArgs e)  {  try  {  double num = Convert.ToDouble(numBuilder);  listNumbers.Add(num);  // store a copy of the number in a temp variable  double temp = num;  if (btnAddClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Add();  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  // clear the list  // add the result and temp value into the list  // res: index 0, temp: index 1  listNumbers.Clear();  listNumbers.Add(res);  listNumbers.Add(temp);  // this will allow user to perform last operation  // using the last number every time equal btn is clicked  }  else if (btnSubClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Sub();  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  listNumbers.Clear();  listNumbers.Add(res);  listNumbers.Add(temp);  }  else if (btnMulClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Mul();  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} \* {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  listNumbers.Clear();  listNumbers.Add(res);  listNumbers.Add(temp);  }  else if (btnDivClicked)  {  calculate.Num1 = listNumbers[0];  calculate.Num2 = listNumbers[1];  res = calculate.Div();  txtCalc.Text = res.ToString();  toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";  toWrite.WriteCalcFile();  listNumbers.Clear();  listNumbers.Add(res);  listNumbers.Add(temp);  }  // change state of equal btn  btnEqualClicked = true;  }  catch (Exception)  {  MessageBox.Show("Error! Please input a number.","Invalid Input");  }    }  private void btnCalcClear\_Click(object sender, EventArgs e)  {  numBuilder = "";  txtCalc.Text = "";  listNumbers.Clear();  btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;  }  private void btnCalcExit\_Click(object sender, EventArgs e)  {  if (MessageBox.Show("Do you want to close this window? ", "Close Calculator", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  }  } |

|  |
| --- |
| **FORM:** IPV4 VALIDATOR **NAME:** frmIPv4Validator |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Text.RegularExpressions;  using System.Net;  namespace WinFormProject  {  public partial class frmIPv4Validator : Form  {  public frmIPv4Validator()  {  InitializeComponent();  }  DateTime openFormTime, closeFormTime;  private void frmIPv4Validator\_Load(object sender, EventArgs e)  {  openFormTime = DateTime.Now;  lblIPValDate.Text = openFormTime.ToLongDateString();  }  private void btnIPVal\_Click(object sender, EventArgs e)  {  Regex validIP = new Regex(@"^(25[0-5]|2[0-4]\d|[0-1]?\d?\d)(\.(25[0-5]|2[0-4]\d|[0-1]?\d?\d)){3}$");  // examples: 192.168.0.10 / 192.186.000  string ipAddress = txtIPVal.Text.Trim();  if (validIP.IsMatch(ipAddress))  {  MessageBox.Show($"{ipAddress}\nThis IP is correct", "Valid IP");  DataStream toWrite = new DataStream();  toWrite.FileName = "BinaryIpv4";  toWrite.MsgBoxTitle = "IPv4 Validator";  toWrite.Output = ipAddress;  toWrite.WriteBinFile();  }  else if (ipAddress.Length == 0)  {  MessageBox.Show("Error! Please input an IPv4 address.", "Invalid IP");  }  else  {  MessageBox.Show($"{ipAddress}\n" +  $"The IP must have 4 bytes\n" +  $"Integer number between 0 to 255\n" +  $"Separated by a dot (255.255.255.255)", "Invalid IP");  }  }  private void btnIPValReset\_Click(object sender, EventArgs e)  {  txtIPVal.Text = "";  }  private void btnIPValExit\_Click(object sender, EventArgs e)  {  closeFormTime = DateTime.Now;  TimeSpan totRunTime = closeFormTime.Subtract(openFormTime);  if (MessageBox.Show("Do you want to close this window?\n" +  $"Total runtime: {totRunTime.Minutes} mins and {totRunTime.Seconds} seconds", "Close IPv4 Validator", MessageBoxButtons.YesNo) == DialogResult.Yes)  {  this.Close();  }  }  }  } |

|  |
| --- |
| **FORM:** DISPLAY READ FILE **NAME:** frmReadFile |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace WinFormProject  {  public partial class frmReadFile : Form  {  public frmReadFile()  {  InitializeComponent();  }  public string fileOutput { get; set; }  public string frmTitle { get; set; }  private void frmReadFile\_Load(object sender, EventArgs e)  {  rtfReadBox.Text = fileOutput;  lblReadTitle.Text = frmTitle;  }  private void btnExit\_Click(object sender, EventArgs e)  {  this.Close();  }  }  } |

1. **Present the classes and/or methods that you create or you did use in the project.**

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
| 1. **DataStream.cs** | This class handles the writing and reading of both text and binary files of all forms |
| 1. public void CreateDir() | Function to check if directory exists. If directory does not exist, create it. |
| 1. public void WriteFile() | Function to write to text file |
| 1. public string ReadFile() | Function to read from text file |
| 1. public void WriteCalcFile() | Function to write the calculator file |
| 1. public void WriteBinFile() | Function to write the binary file |
| 1. **CurrencyEx.cs** | This class handles the calculations for the currency conversions |
| 1. public decimal ConvertToCAD (string currencyFrom) | Function to convert from any currency to Canadian Dollar (CAD) |
| 1. public decimal ConvertToCurrency(decimal convCAD, string currencyTo) | Function to convert from Canadian Dollar (CAD) to any currency |
| 1. **ConvertTemp.cs** | This class handles the computations for temperature conversion and generate the temperature description |
| 1. public decimal ConvertToCelcius() | Function to convert from Fahrenheit (F) to Celsius (C) |
| 1. public decimal ConvertToFarenheit() | Function to convert from Fahrenheit (F) to Celsius (C) |
| 1. public string DescribeTemperature(char unitFrom) | Function to generate the temperature description for display |
| 1. **Calculator.cs** | This class handles all the computations for the operations and management of input list |
| 1. public double Add() | Function to add two numbers |
| 1. public double Sub() | Function to subtract two numbers |
| 1. public double Mul() | Function to multiply two numbers |
| 1. public double Div() | Function to divide two numbers |
| 1. public double Solve (List<double> listNumbers,string btn) | Function to perform calculation based on activated operation and manage the list of numbers |

1. **Present the difficulties that you have, what was the hardest and the easiest part of your project.**

For me, I took the most time building the calculator. The hardest part in doing it was making it work like an actual calculator. I started by creating a code that would handle operations for two numbers, but it got more complicated as I try to handle all possible scenarios like performing multiple operations on multiple numbers. However, the rest of the forms in the project went smoothly. I opted to create separate classes for the forms because I found it more convenient, and it made my code look cleaner and simpler to understand.